Channel Assignment on Strongly-Simplicial Graphs *

A.A. Bertossi
Dept. Comp. Science
University of Bologna, Italy
bertossi@cs.unibo.it

M.C. Pinotti
Dept. Comp. Science & Telecomm.
University of Trento, Italy
pinotti@science.unitn.it

R. Rizzi
Dept. Comp. Science & Telecomm.
University of Trento, Italy
rrizzi@science.unitn.it

Abstract

Given a vector $(\delta_1, \delta_2, \ldots, \delta_t)$ of non increasing positive integers, and an undirected graph $G = (V, E)$, an $L(\delta_1, \delta_2, \ldots, \delta_t)$-coloring of G is a function f from the vertex set V to a set of nonnegative integers such that $|f(u) - f(v)| \geq \delta_i$, if $d(u, v) = i$, $1 \leq i \leq t$, where $d(u, v)$ is the distance (i.e. the minimum number of edges) between the vertices u and v. This paper presents efficient algorithms for finding optimal $L(1, \ldots, 1)$-colorings of trees and interval graphs. Moreover, efficient algorithms are also provided for finding approximate $L(\delta_1, 1, \ldots, 1)$-colorings of trees and interval graphs, as well as approximate $L(\delta_1, \delta_2)$-colorings of unit interval graphs.

1 Introduction

In the *channel assignment* problem for wireless communication networks, the scarce radio spectrum is partitioned into a set of disjoint channels. The same channel can be reused by two stations at the same time provided that no interference arises. The interference phenomena are so strong that even different channels assigned to two near stations must be sufficiently apart in the radio spectrum. To avoid such interference, a *separation vector* $(\delta_1, \delta_2, \ldots, \delta_t)$ of non increasing positive integers is introduced in such a way that channels assigned to interfering stations at distance i be at least δ_i apart, with $1 \leq i \leq t$, while the same channel can be reused only at stations whose distance is larger than t [6]. The purpose of channel assignment algorithms is to assign channels to the stations so that the above channel separations are verified and the difference between the highest and lowest channels assigned is kept as small as possible.

Formally, the channel assignment problem can be modeled as an appropriate coloring problem on an undirected graph $G = (V, E)$ representing the wireless network topology, whose vertices in V correspond to stations, and edges in E correspond to pairs of stations that can hear each other transmission. Specifically, given a vector $(\delta_1, \delta_2, \ldots, \delta_t)$ of non increasing positive integers, and an undirected graph $G = (V, E)$, an $L(\delta_1, \delta_2, \ldots, \delta_t)$-coloring of G is a function f from the vertex set V to the set of nonnegative integers $\{0, \ldots, \lambda\}$ such that $|f(u) - f(v)| \geq \delta_i$, if $d(u, v) = i$, $1 \leq i \leq t$, where $d(u, v)$ is the distance (i.e. the minimum number of edges) between the vertices u and v. An *optimal* $L(\delta_1, \delta_2, \ldots, \delta_t)$-coloring for G is one minimizing λ over all such colorings. Note that, since the set of colors includes 0, the overall number of colors involved by an optimal coloring f is in fact $\lambda + 1$ (although, due to the channel separation constraint, some colors in $\{1, \ldots, \lambda - 1\}$ might not be actually assigned to any vertex). Thus, the channel assignment problem consists of finding an optimal $L(\delta_1, \delta_2, \ldots, \delta_t)$-coloring for G.

The channel assignment problem has been widely studied in the past [1, 2, 3, 4, 5, 7, 9, 10]. This paper further investigates the channel assignment problem for particular separation vectors and two specific classes of graphs – trees and interval graphs.

First, the notions of t-simplicial and strongly-simplicial vertices of a graph will be introduced. Two algorithms will be devised to optimally solve the $L(1, \ldots, 1)$-coloring problem on trees and interval graphs, which run in $O(nt)$ time, where n is the number of vertices. Such algorithms will then be generalized to find approximate solutions for the $L(\delta_1, 1, \ldots, 1)$-coloring problem on the same classes of graphs. Moreover, approximate $L(\delta_1, \delta_2)$-colorings of unit interval graphs are also devised.

2 Preliminaries

Throughout this paper, it is assumed that G is a connected undirected graph with at least 2 vertices and that the separations verify $\delta_1 \geq \delta_2 \geq \ldots \geq \delta_t$.

The $L(1, \ldots, 1)$-coloring problem can be reduced to a classical coloring problem on an *augmented* graph $A_{G,t}$ obtained as follows. The vertex set of $A_{G,t}$ is

*This work is supported by MIUR-RealWine Research Program.

the same as the vertex set of G, while an edge uv belongs to the edge set of $A_{G,t}$ iff the distance $d(u,v)$ between the vertices u and v in G satisfies $d(u,v) \leq t$. The size of the largest clique in $A_{G,t}$ is a lower bound for the $L(1,\ldots,1)$-coloring problem. Clearly, a lower bound for the $L(1,\ldots,1)$-coloring problem is also a lower bound for the $L(\delta_1, 1,\ldots,1)$-coloring problem, with $\delta_1 \geq 1$.

For any value of $t \leq |V|$, let $\lambda^*_{G,t}$ denote the minimum value of λ over all the $L(1,\ldots,1)$-colorings $f : V \rightarrow \{0,\ldots,\lambda\}$ of $G = (V,E)$. Note that $\lambda^*_{G,1} \geq 1$ since G is assumed to be connected and has at least 2 vertices, and that $\lambda^*_{G,t} + 1$ is at least as large as the size of the largest clique of the augmented graph $A_{G,t}$.

Lemma 1 *The largest color needed by any $L(\delta_1, \delta_2, \ldots, \delta_t)$-coloring is at least $max_{1\leq i\leq t}\{\delta_i \lambda^*_{G,i}\}$.*

Given $G = (V,E)$, let S be a subset of V. Then $G[S]$ denotes the subgraph of G induced by S, i.e. $G[S] = (S, \{uv \in E \ : u,v \in S\})$. A vertex x of G is called *t-simplicial* when, for every pair of vertices u and v such that $d(x,u) \leq t$ and $d(x,v) \leq t$, it holds also that $d(u,v) \leq t$. A vertex x is called *strongly-simplicial* when x is t-simplicial for any value of t.

Lemma 2 *Given $G = (V,E)$ and an integer t, let v be a t-simplicial vertex of G. Consider $G' = G[V - \{v\}]$ and let f' be an optimal $L(1,\ldots,1)$-coloring of G' using $\lambda^*_{G',t}$ as the largest color. Define an $L(1,\ldots,1)$-coloring f of V extending f' to v so that $f(x) = min\ \{i : i \neq f'(u)$ for each $u \in G'$ with $d(u,v) \leq t\}$ if $x = v$, and $f(x) = f'(x)$ if $x \in V - \{v\}$. Then f is an optimal $L(1,\ldots,1)$-coloring for G.*

Proof Clearly, $\lambda^*_{G,t} \geq \lambda^*_{G',t}$. If $f(v) \leq \lambda^*_{G',t}$, then f is trivially optimal. Assume therefore that $f(v) > \lambda^*_{G',t}$, and let $N_t(v) = \{u \in V - \{v\} \ : \ d(u,v) \leq t\}$. Then, $\{f'(u) \ : \ u \in N_t(v)\} = \{0,\ldots,\lambda^*_{G',t}\}$. Since v is t-simplicial, any two vertices in $N_t(v) \cup \{v\}$ are at distance at most t, and hence $N_t(v) \cup \{v\}$ is a clique of $A_{G,t}$. Since $|N_t(v) \cup \{v\}| = \lambda^*_{G',t} + 2$, and $f(v) = \lambda^*_{G',t} + 1$, then f is optimal.

Note that verifying whether a vertex is t-simplicial or not can be done in polynomial time. Therefore, Lemma 2 implies the existence of an algorithm that optimally solves in polynomial time the $L(1,\ldots,1)$-coloring problem on any class of graphs closed under taking induced subgraphs and with the property that every graph of that class has a t-simplicial vertex. In this paper, we will look more closely at two classes of graphs with this property: trees and interval graphs.

3 Interval Graphs

A graph $G = (V,E)$ is termed an *interval graph* if it has an *interval representation*, namely, if each vertex of V can be represented by an interval of the real line such that there is an edge $uv \in E$ if and only if the intervals corresponding to u and v intersect.

More formally, let the graph $G = (V,E)$ have n vertices. Two integers l_v and r_v, with $l_v < r_v$, (the *interval endpoints*) are associated to every vertex v of G, and there is an edge $uv \in E$ if and only if $l_u < l_v < r_u$ or $l_u < r_v < r_u$. Without loss of generality, one can assume that all the $2n$ interval endpoints are distinct and are indexed from 1 to $2n$.

Lemma 3 *Every interval graph has a strongly-simplicial vertex.*

Proof Let $G = (V,E)$ be an interval graph with n vertices, and consider its interval representation. It will be now shown that x is t-simplicial for any value of t. Let x be the vertex of G whose left endpoint l_x is maximum. Consider two vertices u and v such that $d(u,x) \leq t$ and $d(v,x) \leq t$. Without loss of generality, let $l_u < l_v < l_x$. Since there is a shortest path $sp(u,x)$ between u and x, there must be a vertex $w \in sp(u,x) - \{x\}$ such that $l_w \leq l_v < r_w$. Therefore, $d(u,v) \leq d(u,w) + 1 \leq d(u,x) \leq t$, and vertex x is t-simplicial. Since such a condition holds for any $t \leq n$, x is strongly-simplicial.

Lemmas 2 and 3 suggest to scan the vertices of an interval graph by increasing left endpoints since, in this way the t-simplicial vertex v of the induced subgraph $G[\{1,\ldots,v\}]$ is processed at every time, for $1 \leq v \leq n$.

3.1 Optimal $L(1,\ldots,1)$-coloring

In this subsection, an $O(nt)$ time algorithm is proposed to find an optimal $L(1,\ldots,1)$-coloring of interval graphs, which exploits the properties given in Lemmas 2 and 3.

Consider the interval representation of G, and assume that the intervals (vertices) are indexed by increasing left endpoints, namely $l_1 < l_2 < \ldots < l_n$. For each endpoint k, an interval v is called *open* if $l_v \leq k < r_v$ and *deepest* if it is open and its right endpoint is maximum.

The algorithm scans the $2n$ interval endpoints from left to right, and it maintains a family of $t+1$ sets of colors, called *palettes*, denoted by $P_0, P_1, \ldots, P_t$. For each endpoint k, the palette P_0 contains the readily usable colors, while the palette P_t includes the colors used for the currently open intervals. Moreover, the

Algorithm Interval-$L(1,\ldots,1)$-coloring (G,t);

```
set L_v := ∅ for every v ∈ V;
set P_i := ∅ for i = 0,...,t; MAX-r := 0; λ*_{G,t} := -1;
for k := 1 to 2n do
   if k = l_v for some v, then
      if P_0 = ∅ then
         λ*_{G,t} := λ*_{G,t} + 1;
         insert λ*_{G,t} in P_0;
      extract a color c from P_0 and set f(v) := c;
      insert color c into both L_v and P_t;
      if r_v > MAX-r then
         MAX-r := r_v;
         DEEP := v;
   otherwise, if k = r_v for some v, then
      for each color c in L_v do
         let j be such that c ∈ P_j;
         extract c from P_j and insert c into P_{j-1};
         if j > 1 then insert c into L_deep;
```

Figure 1. The algorithm for optimal $L(1,\ldots,1)$-coloring an interval graph $G=(V,E)$.

generic palette P_i, with $1 \leq i \leq t-1$, contains the colors that can be reused as soon as all the next i deepest intervals will be ended.

Whenever a new interval v begins, that is a left endpoint l_v is encountered, v is colored by a color c extracted from the palette P_0 and, if needed, the deepest interval is updated. Moreover, the used color c is put both in the palette P_t and in the set L_v of colors depending on vertex v.

Whenever an interval v ends, that is a right endpoint r_v is encountered, every color c belonging to L_v is moved from its current palette, say P_j, to the previous palette P_{j-1} and it is inserted in the set L_{deep} of the colors depending on the current deepest interval DEEP.

Figure 1 illustrates the algorithm for optimal $L(1,\ldots,1)$-coloring of interval graphs.

Lemma 4 *Consider to be at the beginning of iteration k of the algorithm. Let u be any vertex yet uncolored and let w be any vertex colored with some color c. Assume $c \in P_j$. Let z be the vertex with r_z maximum and such that $c \in L_z$. Then the following holds:*

(i) $d(u,w) > t-j$;

(ii) *if* $l_u > r_z$, *then* $d(u,w) > t-j+1$;

(iii) *if* $l_u < r_z$, *then* $d(u,w) = t-j+1$.

(iv) *the minimum distance from z to a vertex colored c is $t-j$.*

Proof The statement is vacuously true before the first iteration, when every vertex is still uncolored. Assume that the statement holds before iteration k, and let us show that it holds also after iteration k. There are two cases to consider, depending on whether k corresponds to a left or right interval endpoint.

Case 1: At iteration k, $k = l_v$ for some vertex v.

In this case, it is enough to check what happens for $w = v$. Indeed, during iteration k, v is the only vertex which gets assigned a color, namely c, which is inserted both into P_t and L_v. Moreover, no other color moves to a palette of lower index. Hence, $j = t$ and $z = v = w$. Since u is uncolored whereas v has just been colored, then $u \neq v$ which accounts for (i): $d(u,v) > 0 = t-t$. Clearly, if $l_u > r_z$, or in other words $l_u > r_w$, then $d(u,w) > 1$, which gives (ii). Moreover, if $l_u < r_z$, or in other words $l_u < r_w$, then $d(u,w) = 1$, which gives (iii). Finally, (iv) is trivial.

Case 2: At iteration k, $k = r_v$ for some vertex v.

Here one needs only to check what happens when $c \in L_v$. Now, since $c \in L_v$ and $l_u > r_v$, then $d(u,w) > t-j$ follows since (ii) was true at the beginning of iteration k. Furthermore, in case $j > 0$, then $c \in L_{\text{deep}}$. If $l_u > r_{\text{deep}}$, then $d(u,v) > t-j+1$, since in any path from u to v the vertex closest to u must be uncolored. Moreover, (ii) follows from the fact that, clearly, $z = \text{deep}$. Besides, if $l_u < r_z$, or equivalently $l_u < r_{\text{deep}}$, then $d(u,w) = t-j+1$ follows since (iv) was true at the beginning of iteration k. Finally, $d(\text{deep}, v) = 1$, which combined with (i) gives (iv).

Theorem 1 *The Interval-$L(1,\ldots,1)$-coloring algorithm gives an optimal coloring and runs in $O(nt)$ time.*

Proof The correctness follows from Lemmas 2, 3, and 4.

All the palettes P_i, $0 \leq i \leq t$ and all the sets L_u, with $1 \leq u \leq n$, are implemented by double linked lists, so that insertions and extractions can be performed in constant time by means of a vector C, indexed by colors, where each entry $C[c]$ stores the current palette index j, to which c belongs, along with a pointer to the position of c within the double linked list P_j.

In order to evaluate the overall time complexity, observe that the algorithm consists of $2n$ iterations and, at every iteration k, each step takes $O(1)$ time, except the scan of list L_v whenever interval v ends. Each color c, after being assigned to a vertex, goes through t lists L_{deep} before to be reassigned to another vertex. In fact, every time c moves from palette P_j to P_{j-1}, the distance between the last vertex colored c and the uncolored vertices increases by one as shown in Lemma 4. Hence, between two consecutive assignments of the same color c, there are at most $t+1$ moves, each performed in a different iteration and each taking constant time. Let m_c be the overall number of vertices of G colored c. Therefore, the total number of moves for color c is at most $(t+1)m_c$. Summing up over all the used colors, the overall number of moves is at most $\sum_c (t+1)m_c = O(nt)$, since $\sum_c m_c = n$. In conclusion, the algorithm takes $O(nt)$ time provided that the interval representation of G is available and the $2n$ interval endpoints are sorted.

3.2 Approximate $L(\delta_1, 1, \ldots, 1)$-coloring

In this subsection, a generalization of the Interval-$L(1,\ldots,1)$-coloring algorithmn is proposed to find an approximate $L(\delta_1, 1, \ldots, 1)$-coloring.

At first, the algorithm computes $\lambda^*_{G,1}$ and $\lambda^*_{G,t}$ invoking twice the Interval-$L(1,\ldots,1)$-coloring algorithm. As before, the palettes $P_0, P_1, \ldots, P_t$ are maintained, but P_0 is initialized to the set of colors $\{0, 1, \ldots, U\}$, where $U = \lambda^*_{G,t} + 2(\delta_1 - 1)\lambda^*_{G,1}$.

The main difference, with respect to the Interval-$L(1,\ldots,1)$-coloring algorithm, relies on the fact that when a color c is extracted from P_0 and assigned to vertex v, the δ_1-separation constraint must be guaranteed. Therefore, all the colors in

$$\{\max\{0, c-\delta_1+1\}, \ldots, \min\{c+\delta_1-1, U\}\},$$

but c itself, are inserted in P_1. In this way, such colors cannot be reused until the interval v ends.

Theorem 2 *The Interval-$L(\delta_1, 1, \ldots, 1)$-coloring algorithm gives a 3-approximate coloring using $\lambda^*_{G,t} + 2(\delta_1 - 1)\lambda^*_{G,1}$ as the largest color.*

Proof The correctness follows from Theorem 1 and from the fact that, once an interval is colored c, the δ_1-separation constraint is achieved by inserting in P_1 the $2(\delta_1 - 1)$ closest colors to c.

One also needs to show that the colors initially in P_0 are enough to obtain a legal $L(\delta_1, 1, \ldots, 1)$-coloring of G. When a color c is assigned to an interval v, all intervals in $G[\{1, \ldots, v\}]$ at distance smaller than or equal to t from v must get a different color since v is t-simplicial. Hence, all their colors, which are at most $\lambda^*_{G,t}$, must belong to $P_1 \cup P_2 \cup \ldots \cup P_t$. Moreover, due to the δ_1 separation-constraint, at most $2(\delta_1 - 1)\lambda^*_{G,1}$ colors have been forced into P_1. Hence, $|P_1 \cup P_2 \cup \ldots \cup P_t| \le \lambda^*_{G,t} + 2(\delta_1 - 1)\lambda^*_{G,1}$ and since initially there were $\lambda^*_{G,t} + 2(\delta_1 - 1)\lambda^*_{G,1} + 1$ colors in P_0, it holds $|P_0| - |P_1 \cup P_2 \cup \ldots \cup P_t| \ge 1$. Thus, there is always an available color that can be assigned to v.

In order to find the approximation factor, the ratio between the upper bound U on the maximum color used by the above algorithm and the lower bound, $L = \max\left\{\delta_1\lambda^*_{G,1}, \lambda^*_{G,t}\right\}$ on the maximum color needed, given by Lemma 1, is

$$\frac{U}{L} = \frac{\lambda^*_{G,t} + 2(\delta_1 - 1)\lambda^*_{G,1}}{\max\left\{\delta_1\lambda^*_{G,1}, \lambda^*_{G,t}\right\}}.$$

If $\delta_1\lambda^*_{G,1} \ge \lambda^*_{G,t}$, the above ratio U/L becomes

$$\frac{U}{L} = \frac{\lambda^*_{G,t} + 2(\delta_1 - 1)\lambda^*_{G,1}}{\delta_1\lambda^*_{G,1}} \le$$

$$\frac{3\delta_1\lambda^*_{G,1} - 2\lambda^*_{G,1}}{\delta_1\lambda^*_{G,1}} \le 3 - \frac{2}{\delta_1} \le 3.$$

If $\delta_1\lambda^*_{G,1} < \lambda^*_{G,t}$, the ratio U/L is

$$\frac{U}{L} = \frac{\lambda^*_{G,t} + 2(\delta_1 - 1)\lambda^*_{G,1}}{\lambda^*_{G,t}} \le$$

$$\frac{3\lambda^*_{G,t} - 2\lambda^*_{G,1}}{\lambda^*_{G,t}} \le 3 - 2\frac{\lambda^*_{G,1}}{\lambda^*_{G,t}} \le 3.$$

3.3 Approximate $L(\delta_1, \delta_2)$-coloring of unit interval graphs

This subsection deals with the $L(\delta_1, \delta_2)$-coloring problem on the class of *unit interval graphs*. This is a subclass of the interval graphs for which all the intervals are of the same length, or equivalently for which no interval is properly contained within another. Recalling that vertices are assumed to be indexed by increasing left endpoints, the main property of unit interval graphs is that whenever $v < u$ and $vu \in E$, then the vertex set $\{v, v+1, \ldots, u-1, u\}$ forms a clique and $u \le v + \lambda^*_{G,1}$ (as a consequence, the maximum vertex w at distance 2 from v verifies $w \le v + 2\lambda^*_{G,1}$).

In this subsection, it is assumed that the unit interval graph to be colored is not a path, since otherwise the optimal $L(\delta_1, \delta_2)$-coloring algorithm in [10] can be applied.

In Figure 2, a linear time algorithm, called Unit-Interval-$L(\delta_1, \delta_2)$-coloring, is presented. The algorithm distinguishes two cases and uses either at most δ_2 additional colors with respect to the optimum, when $\delta_1 > 2\delta_2$, or at most $2\delta_2$ additional colors when $\delta_1 \le 2\delta_2$.

Algorithm Unit-Interval-$L(\delta_1,\delta_2)$-coloring (G);

```
if δ1 > 2δ2 then
    for every vertex v ∈ V do
        set p := (v − 1) mod (2λ*_{G,1} + 2);
        if 0 ≤ p ≤ λ*_{G,1}
            then f(v) := δ1(λ*_{G,1} − p)
            else f(v) := δ1(λ*_{G,1} − p) + δ2;
if δ1 ≤ 2δ2 then
    for every vertex v ∈ V do
        f(v) := (2δ2(v − 1)) mod (2δ2λ*_{G,1} + 3δ2);
```

Figure 2. The $L(\delta_1,\delta_2)$-coloring algorithm for a unit interval graph $G=(V,E)$.

Theorem 3 *The Unit-Interval-$L(\delta_1,\delta_2)$-coloring algorithm gives an approximate coloring using as the largest color $\delta_1\lambda^*_{G,1}+\delta_2$, if $\delta_1>2\delta_2$, or $2\delta_2\lambda^*_{G,1}+3\delta_2$, if $\delta_1\le 2\delta_2$.*

Proof When $\delta_1<2\delta_2$, the algorithm colors the vertices by repeating the following sequence of length $2\lambda^*_{G,1}+2$:

$$0,\delta_1,2\delta_1,\ldots,\lambda^*_{G,1}\delta_1,\delta_2,\delta_1+\delta_2,\delta_1+2\delta_2,\ldots,\lambda^*_{G,1}\delta_1+\delta_2.$$

Consider a vertex v colored $c=j\delta_1$, with $0\le j\le\lambda^*_{G,1}$ (an analogous reasoning holds when $c=j\delta_1+\delta_2$). First of all, the color c is used exactly once within the sequence. Thus if c is assigned to vertex v, then it is reused at vertex $v+2\lambda^*_{G,1}+2$, which is at distance at least 3 from v, since otherwise $\lambda^*_{G,1}$ would not be optimal. In order to verify the δ_1 separation-constraint, it remains to check that all the colors $c-\delta_1+1,\ldots,c-1,c+1,\ldots,c+\delta_1-1$ cannot be reused for any vertex at distance 1 from v. Among such colors, only the color $c+\delta_2$ is used in the sequence, and it is assigned to the vertices $v\pm(\lambda^*_{G,1}+1)$, as one can easily check by inspecting the sequence above. The vertices v and $v\pm(\lambda^*_{G,1}+1)$ cannot be adjacent, since otherwise there would be a clique of size $\lambda^*_{G,1}+2$, including vertices $v,v\pm 1,\ldots,v\pm(\lambda^*_{G,1}+1)$, which contradicts the optimality of $\lambda^*_{G,1}$. Moreover, the δ_2 separation-constraint trivially follows from the fact that the colors $c-\delta_2+1,\ldots,c-1,c+1,\ldots,c+\delta_2-1$ are never used.

When $\delta_1\le 2\delta_2$, the algorithm colors the vertices by repeating the following sequence of length $2\lambda^*_{G,1}+3$:

$$0,2\delta_2,4\delta_2,\ldots,2(\lambda^*_{G,1}+1)\delta_2,\delta_2,3\delta_2,5\delta_2,\ldots,2\lambda^*_{G,1}\delta_2+\delta_2.$$

Consider, again, a vertex v colored $f(v)=c$. As in the previous case, the color c is used exactly once within the sequence. Thus if c is assigned to vertex v, then it is reused at vertex $v+2\lambda^*_{G,1}+3$, which cannot be at distance 1 or 2 from v. As regard to the δ_1 separation-constraint, note that, other than c, only the colors $c\pm\delta_2$ are used in the sequence. They are assigned to vertices $v\mp(\lambda^*_{G,1}+1)$, as one can easily check by computing $f(v\mp(\lambda^*_{G,1}+1))$. Those vertices cannot be adjacent because otherwise the vertices $v,v\mp 1,\ldots,v\mp(\lambda^*_{G,1}+1)$ would form a clique, contradicting the optimality of $\lambda^*_{G,1}$. Furthermore, the colors $c-\delta_2+1,\ldots,c-1,c+1,\ldots,c+\delta_2-1$ are never used, and hence the δ_2 separation-constraint holds too.

In order to find the approximation factor, observe that, by Lemma 1, the largest color used by any $L(\delta_1,\delta_2)$-coloring is at least $L=\max\{\delta_1\lambda^*_{G,1},\delta_2\lambda^*_{G,2}\}$. When $\delta_1>2\delta_2$, L becomes $\delta_1\lambda^*_{G,1}$ since $\delta_2\lambda^*_{G,2}\le 2\delta_2\lambda^*_{G,1}<\delta_1\lambda^*_{G,1}$. In contrast, when $\delta_1\le 2\delta_2$, L can be either $\delta_1\lambda^*_{G,1}$ or $\delta_2\lambda^*_{G,2}$. On the other hand, the maximum color U used by the algorithm is $\delta_1\lambda^*_{G,1}+\delta_2$ when $\delta_1>2\delta_2$, and $2\delta_2\lambda^*_{G,1}+3\delta_2$ when $\delta_1\le 2\delta_2$.

Therefore, it holds:

$$\frac{U}{L}=\begin{cases}\frac{\delta_1\lambda^*_{G,1}+\delta_2}{\delta_1\lambda^*_{G,1}} & \text{if } \delta_1>2\delta_2,\\[2ex] \frac{2\delta_2\lambda^*_{G,1}+2\delta_2}{\max\{\delta_1\lambda^*_{G,1},\delta_2\lambda^*_{G,2}\}} & \text{if } \delta_1\le 2\delta_2.\end{cases}$$

Although the ratio U/L can be evaluated exactly since the values of $\lambda^*_{G,1}$ and $\lambda^*_{G,2}$ can be computed in polynomial time invoking the Interval-$L(1,\ldots,1)$-coloring algorithm, the worst value U/L can be bounded from above by a constant, independent of G, δ_1 and δ_2.

Specifically, when $\delta_1>2\delta_2$, the ratio is

$$\frac{\delta_1\lambda^*_{G,1}+\delta_2}{\delta_1\lambda^*_{G,1}}=1+\frac{\delta_2}{\delta_1}\frac{1}{\lambda^*_{G,1}}\le\frac{3}{2},$$

because $\frac{\delta_2}{\delta_1}\frac{1}{\lambda^*_{G,1}}\le\frac{1}{2}$ from the assumption that the unit interval graph is connected.

Moreover, when $\delta_1\le 2\delta_2$ and $L=\delta_1\lambda^*_{G,1}$, the above ratio becomes

$$\frac{2\delta_2\lambda^*_{G,1}+2\delta_2}{\delta_1\lambda^*_{G,1}}=2\frac{\delta_2}{\delta_1}\left(1+\frac{1}{\lambda^*_{G,1}}\right)\le 3,$$

since $\frac{\delta_2}{\delta_1}\le 1$ and $\frac{1}{\lambda^*_{G,1}}\le\frac{1}{2}$ from the assumption that the unit interval graph is not a path.

Finally, when $\delta_1\le 2\delta_2$ and $L=\delta_2\lambda^*_{G,2}$,

$$\frac{2\delta_2\lambda^*_{G,1}+2\delta_2}{\delta_2\lambda^*_{G,2}}=2\frac{\lambda^*_{G,1}}{\lambda^*_{G,2}}\left(1+\frac{1}{\lambda^*_{G,1}}\right)\le 3,$$

since $\frac{\lambda^*_{G,1}}{\lambda^*_{G,2}} \leq 1$ and $\frac{1}{\lambda^*_{G,1}} \leq \frac{1}{2}$ as in the previous case.

In conclusion, the Unit-Interval-$L(\delta_1, \delta_2)$-coloring algorithm gives an approximate solution which, in the worst case, is far from the optimal by a factor of 3.

It is worth to note that, when $\delta_1 = 2$ and $\delta_2 = 1$, the ratio U/L becomes $\frac{2\lambda^*_{G,1}+2}{2\lambda^*_{G,1}}$, namely the same derived in [8] for the $L(2,1)$-coloring problem on unit interval graphs.

4 Trees

Given an ordered tree T of height h and an integer $\ell \leq h$, the *induced subtree* T_ℓ consists of all the vertices v with level $l(v) \leq \ell$ as well as all the original edges among them. For each vertex v of T, let $anc_i(v)$ denote the ancestor of v at distance i from v (which clearly is at level $l(v) - i$). Besides, let $D_i(v)$ denote the set of the *descendents* of v at distance i from v (which clearly belong to level $l(v) + i$).

Lemma 5 *Every tree has a strongly-simplicial vertex.*

Proof Let T be a tree and consider a vertex x with $l(x) = h$. Let t be any arbitrary integer not larger than $2h$. Consider two vertices u and v such that $d(u,x) \leq t$ and $d(v,x) \leq t$. Consider also the shortest paths $sp(x,v)$, $sp(x,u)$, and $sp(x,w)$, where w is the vertex of smallest level belonging to both $sp(x,v)$ and $sp(x,u)$. Since $l(x) \geq \max\{l(u), l(v)\}$, then $\min\{d(u,w), d(v,w)\} \leq d(x,w)$. Assume w.l.o.g. $d(u,w)$ to be minimum. Then, $d(u,v) = d(u,w) + d(w,v) \leq d(x,w) + d(w,v) \leq d(x,v) \leq t$. Therefore, vertex x is t-simplicial. Since such a condition holds for any $t \leq 2h$, x is strongly-simplicial.

Lemmas 2 and 5 suggest to visit the tree in breadth-first-search order, namely scanning the vertices by increasing levels, and those at the same level from left to right. In this way, at every turn, a t-simplicial vertex v at level $l(v)$ of the induced subtree $T_{l(v)}$ is processed, for $1 \leq v \leq n$. For this purpose, hereafter, it is assumed that the vertices are numbered according to the above breadth-first-search order.

4.1 Optimal $L(1,\ldots,1)$-coloring

In this subsection, an $O(nt)$ time algorithm is exhibited to find an optimal $L(1,\ldots,1)$-coloring of trees. The algorithm first performs a preprocessing in order to compute, for each vertex x of T, the $t+1$ lists D_i, $0 \leq i \leq t$. Such a computation can be performed in $O(nt)$ time by visiting the tree in postorder. The corresponding recursive procedure, called *Explore-Descendents*, is shown in Figure 3 and has to be invoked starting from the root r of T. It can be easily modified to compute in the same $O(nt)$ time also all the cardinalities $|D_i(x)|$ for every vertex x simply by substituting the last statement with $|D_i(x)| := |D_i(x)| + |D_{i-1}(x)|$.

The algorithm also uses another function, called *Up-Neighborhood* and illustrated in Figure 4, which takes in input a vertex y and a distance *uplevel* and returns the set F of the vertices at distance no larger than *uplevel* from vertex y in the induced subtree $T_{l(y)}$, rooted at y. Conceptually, F corresponds to the set $F_{uplevel}(y)$ of vertices up to distance *uplevel* traversed by a Breadth-

Procedure Explore-Descendents $(x, T=(V,E), t)$;

set $D_0(x) := \{x\}$ and $D_i(x) := \emptyset$, for $1 \leq i \leq t$;
if x is not a leaf then
    for every child v of x do
        Explore-Descendents(v, T, t);
        for $i := 1$ to t do
            $D_i(x) := D_i(x) \cup D_{i-1}(v)$;

Figure 3. The recursive procedure to compute the lists of the descendents up to distance t.

Function Up-Neighborhood $(y, uplevel)$: vertex-set;

set $F := \emptyset$;
set $anc := y$;
for $i := 1$ to $\lceil \frac{t}{2} \rceil - 1$ do
    set $anc := father(anc)$;
for $i := \lceil \frac{t}{2} \rceil$ to $uplevel$ do
    set $anc := father(anc)$;
    if $i = \lceil \frac{t}{2} \rceil$ then
        $F := F \cup D_{t-i-1}(anc)$;
        if t is odd then $F := F \cup D_{t-i}(anc)$;
    if $\lceil \frac{t}{2} \rceil < i < t$ then
        $F := F \cup D_{t-i-1}(anc) \cup D_{t-i}(anc)$;
    if $i = t$ then
        $F := F \cup D_{t-i}(anc)$;
return F;

Figure 4. The function to compute the neighborhood of vertex y at distance no larger than *uplevel* in $T_{l(y)}$ rooted at y.

First-Search starting from y in $T_{l(y)}$. However, computing F in such a way would imply to change the root of the subtree at every time the function is invoked. To avoid this, the tree is maintained with the original root, and the set F is computed from the sets of the descendents which have been obtained once for all by the Explore-Descendents procedure.

In practice, when $uplevel = t$, the function visits the vertices along the path from $anc_{\lceil \frac{t}{2} \rceil}(y)$ up to $anc_t(y)$, building the required neighborhood set $F_t(y)$ including the following descendent sets of the vertices on such a path. In details, let $S = D_0(anc_{t-1}(y)) \cup D_1(anc_{t-1}(y)) \cup \ldots \cup D_{\lfloor \frac{t}{2} \rfloor -2}(anc_{\lceil \frac{t}{2} \rceil +1}(y)) \cup D_{\lfloor \frac{t}{2} \rfloor -1}(anc_{\lceil \frac{t}{2} \rceil +1}(y))$. Then, $F_t(y) = D_0(anc_t(y)) \cup S \cup D_{\lfloor \frac{t}{2} \rfloor -1}(anc_{\lceil \frac{t}{2} \rceil}(y))$ if t is even, and $F_t(y) = D_0(anc_t(y)) \cup S \cup D_{\lfloor \frac{t}{2} \rfloor -1}(anc_{\lceil \frac{t}{2} \rceil}(y)) \cup D_{\lfloor \frac{t}{2} \rfloor}(anc_{\lceil \frac{t}{2} \rceil}(y))$ if t is odd.

Note that if $uplevel < t$, the path from $anc_{\lceil \frac{t}{2} \rceil}(y)$ to $anc_{uplevel}(y)$ ends at the root of T and only the sets of descendents associated with those vertices are included.

Clearly, the largest color $\lambda^*_{T,t}$ needed by any $L(1,\ldots,1)$-coloring of T is given by the size of the largest neighborhood set returned by the Up-Neighborhood function. Therefore, $\lambda^*_{T,t} = \max_{y \in V}\{|F_t(y)|\}$, which can be computed in $O(nt)$ time simply modifying the Up-Neighborhood function so as to manipulate the set cardinalities in place of the sets themselves. The Tree-$L(1,\ldots,1)$-coloring algorithm, illustrated in Figure 5, starts invoking the Explore-Descendents procedure to compute the sets of the descendents and the largest color $\lambda^*_{T,t}$, and henceforth initializes accordingly the palette P. Then, observed that all the vertices in the uppermost $\lfloor \frac{t}{2} \rfloor + 1$ levels are all mutually at distance at most t, the algorithm colors them with all different colors extracted from P. The rest of the tree is colored level by level. For each level $\ell > \lfloor \frac{t}{2} \rfloor$, P is set to $\{0,\ldots,\lambda^*_{T,t}\}$. The vertices in level ℓ are then colored from left to right, identifying groups of consecutive vertices which share the same palette. Each group contains all the vertices such that the level of their lowest common ancestor lca is larger than or equal to $\ell - \lceil \frac{t}{2} \rceil + 1$. Each group is identified by its leftmost vertex x and its rightmost vertex $last_x$. When a new level ℓ starts, x is simply the leftmost vertex in level ℓ. Otherwise, two consecutive groups $\{old_x,\ldots,last_{old_x}\}$ and $\{x,\ldots,last_x\}$ at the same level ℓ verify $x = last_{old_x} + 1$. To color the leftmost group, the Up-Neighborhood function is invoked with $uplevel = \min\{t,\ell\}$. Such a procedure returns in F the vertices whose colors are no longer available, and which are then removed from the palette P. Then the colors in P are used to color all the

---

Algorithm Tree-$L(1,\ldots,1)$-coloring $(T = (V,E), t)$;

$\texttt{Explore-Descendents}(r,T,t)$;
set $\lambda^*_{T,t} := \max_{x \in V}\{|F_t(x)|\}$;
set $P := \{0,\ldots,\lambda^*_{T,t}\}$;
for $\ell := 0$ to $\lfloor \frac{t}{2} \rfloor$ do
  for each vertex x with $l(x) = \ell$ do
    extract a color c from P and set $f(x) := c$;
for $\ell := \lfloor \frac{t}{2} \rfloor + 1$ to h do
  set $P := \{0,\ldots,\lambda^*_{T,t}\}$;
  set x to the leftmost vertex with $l(x) = \ell$;
  set $last_x :=$ rightmost vertex in $D_{\lceil \frac{t}{2} \rceil}(anc_{\lceil \frac{t}{2} \rceil}(y))$;
  set $uplevel$ to $\min\{t,\ell\}$;
  set $F := \texttt{Up-Neighborhood}(x, uplevel)$;
  set $P := P - \{c : c = f(u),\ u \in F\}$;
  for each vertex u with $x \le u \le last_x$ do
    extract a color c from P and set $f(u) := c$;
  while $l(last_x + 1) = \ell$ do
    set $old_x := x$, $x := last_x + 1$;
    set $lca :=$ lowest common ancestor(x, old_x);
    set $uplevel$ to $\min\{t, \ell - l(lca) - 1\}$;
    set $F := \texttt{Up-Neighborhood}(old_x, uplevel)$;
    set $P := P \cup \{c : c = f(u),\ u \in F\}$;
    set $F := \texttt{Up-Neighborhood}(x, uplevel)$;
    set $P := P - \{c : c = f(u),\ u \in F\}$;
    for each vertex u with $x \le u \le last_x$ do
      extract a color c from P and set $f(u) := c$;

Figure 5. The algorithm for optimal $L(1,\ldots,1)$-coloring of trees.

vertices in $\{x,\ldots,last_x\}$. For each of the remaining groups at the same level, the palette is updated by invoking twice the Up-Neighborhood function. The first call $\texttt{Up-Neighborhood}(old_x, uplevel)$ returns in F the set $F_{uplevel}(old_x)$ of the vertices whose colors are again available. Such colors are then added to the palette P. The second call $\texttt{Up-Neighborhood}(x, uplevel)$ returns in F the set $F_{uplevel}(x)$ of the vertices whose colors are forbidden, which are then removed from the palette P. As before the colors currently in P are then used to color the vertices of the group.

Theorem 4 *The Tree-$L(1,\ldots,1)$-coloring algorithm gives an optimal coloring and runs in $O(nt)$ time.*

Proof The correctness follows from Lemma 5, while the optimality follows since only the colors in $\{0,\ldots,\lambda^*_{T,t}\}$ are used, and $\lambda^*_{T,t}$ is a lower bound.

The palette P is implemented by means of a double linked list of colors and a vector C indexed by colors, as explained in Theorem 1.

To evaluate the time complexity, observe first that both the computations of $last_x$ and lca require $O(t)$ time. Then, given a vertex v, consider a level ℓ along with the leftmost group at level ℓ for which v belongs to some neighborhood set. For the same level ℓ, once v enters in a neighborhood, it remains in the neighborhood of the leftmost vertex for some consecutive groups, and finally it leaves. Thus, v is involved twice by the Up-Neighborhood function: the first time to be inserted in F and the second time to be removed from F.

Since each vertex v can appear in sets F's only for t consecutive levels $l(v), \ldots, l(v)+t$, v is involved in exactly t insertions into the neighborhood F's and exactly t deletions from the neighborhood F's while coloring the entire tree, and an overall $O(nt)$ time complexity follows.

4.2 Approximate $L(\delta_1, 1, \ldots, 1)$-coloring

In this subsection, a generalization of the Tree-$L(1,\ldots,1)$-coloring algorithm is proposed to find an approximate $L(\delta_1, 1, \ldots, 1)$-coloring. The algorithm computes $\lambda^*_{T,t}$ by means of the Tree-$L(1,\ldots,1)$-coloring algorithm and uses an enriched palette $P = \{0, 1, \ldots, U\}$, where $U = \lambda^*_{T,t} + 2(\delta_1 - 1)$. Moreover, in order to satisfy the δ_1 separation constraint, the instructions to color a vertex u become:

---

find a color c in P such that $|c - f(anc_1(u))| \geq \delta_1$;
extract c from P and set $f(u) := c$;

Therefore, to color a single vertex, $O(\delta_1)$ time is required and hence an overall time complexity of $O(n(t + \delta_1))$ results.

Theorem 5 *The Tree-$L(\delta_1, 1, \ldots, 1)$-coloring algorithm gives a 3-approximate coloring using $\lambda^*_{T,t} + 2(\delta_1 - 1)$ as the largest color.*

Proof The proof is similar to that of Theorem 2.

5 Conclusion

This paper has considered the channel assignment problem for particular separation vectors and two specific classes of graphs – trees and interval graphs. Based on the notions of t-simplicial and strongly-simplicial vertices, $O(nt)$ time algorithms have been proposed to find optimal $L(1,\ldots,1)$-colorings on trees and interval graphs. Such algorithms have been generalized to find approximate $L(\delta_1, 1, \ldots, 1)$-colorings on the same classes of graphs. Moreover, an approximate $L(\delta_1, \delta_2)$-coloring of unit interval graphs has been presented.

Several questions still remain open. For instance, one could devise polynomial time algorithms for finding optimal $L(\delta_1, 1, \ldots, 1)$-colorings of interval graphs and trees, as well as optimal $L(\delta_1, \delta_2)$-colorings of unit interval graphs.

Moreover, one could search for further classes of graphs that verify the strongly-simplicial property.

References

[1] R. Battiti, A.A. Bertossi, and M.A. Bonuccelli, "Assigning Codes in Wireless Networks: Bounds and Scaling Properties", *Wireless Networks*, Vol. 5, 1999, pp. 195-209.

[2] A.A. Bertossi and M.C. Pinotti, "Mappings for Conflict-Free Access of Paths in Bidimensional Arrays, Circular Lists, and Complete Trees", *Journal of Parallel and Distributed Computing*, Vol. 62, 2002, pp. 1314-1333.

[3] A.A. Bertossi, M.C. Pinotti, and R. Tan, "Efficient Use of Radio Spectrum in Wireless Networks with Channel Separation between Close Stations", *DIAL M for Mobility; Int'l ACM Workshop on Discrete Algorithms and Methods for Mobile Computing*, Boston, August 2000.

[4] A.A. Bertossi, M.C. Pinotti, and R.B. Tan, "Channel Assignment with Separation for Interference Avoidance in Wireless Networks", to appear *IEEE Transactions on Parallel and Distributed Systems*

[5] I. Chlamtac and S.S. Pinter, "Distributed Nodes Organizations Algorithm for Channel Access in a Multihop Dynamic Radio Network", *IEEE Transactions on Computers*, Vol. 36, 1987, pp. 728-737.

[6] W.K. Hale, "Frequency Assignment: Theory and Application", *Proceedings of the IEEE*, Vol. 68, 1980, pp. 1497-1514.

[7] S.T. McCormick, "Optimal Approximation of Sparse Hessians and its Equivalence to a Graph Coloring Problem", *Mathematical Programming,* Vol. 26, 1983, pp. 153–171.

[8] D. Sakai, "Labeling Chordal Graphs: Distance Two Condition", *SIAM Journal on Discrete Mathematics*, Vol. 7, 1994, pp. 133-140.

[9] A. Sen, T. Roxborough, and S. Medidi, "Upper and Lower Bounds of a Class of Channel Assignment Problems in Cellular Networks", Technical Report, Arizona State University, 1997

[10] J. Van den Heuvel, R. A. Leese, and M.A. Shepherd, "Graph Labelling and Radio Channel Assignment", *Journal of Graph Theory* Vol. 29, 1998, pp. 263-283.